OWASP TOP 10

Hai Hoang

NASH TECH
The Power to Innovate

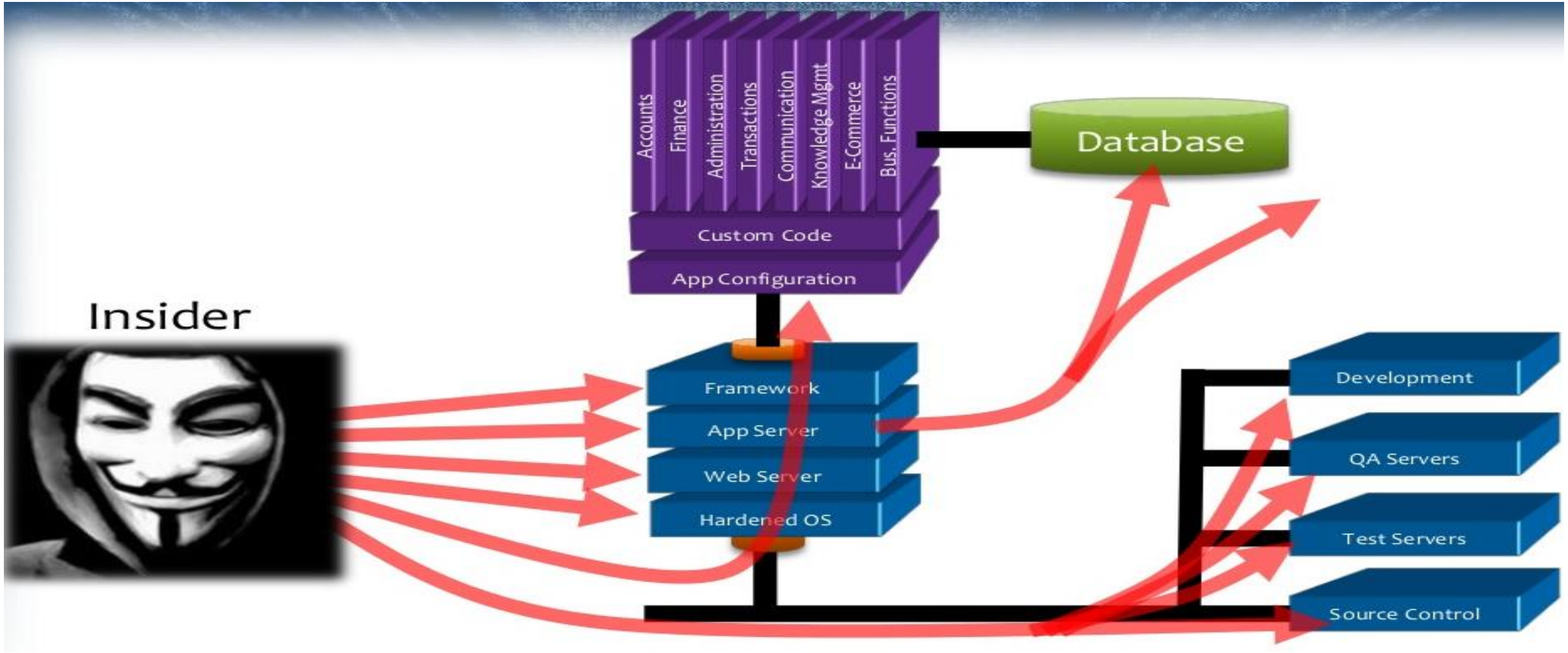
OWASP TOP 10

- A10: Un-validated Redirects and Forwards
- A9: Using Known Vulnerable Components
- A8: Cross-Site Request Forgery (CSRF)
- A7: Missing Function Level Action Control
- A6: Sensitive Data Exposure
- **A5: Security Misconfiguration**

A5: Risks

- **Web Application** rely on a **secure foundation**
  - The **OS** up through the **App Server**.
  - **All the libraries** you are using.
- **Your source codes** are **secret**?
  - **All the places** your source code goes.
  - **Not require** secret source code.
- Do you change **all credentials regularly** in your **production** environment?

A5: Attack Scenarios



A5: Impact

- **Unauthorized access** to **system data** or **functionality**.
- **Data** could be **stolen** or **modified** slowly over time.

A5: Prevent (*)

- **Verify your system’s configuration management**
- **Deactivate unnecessary stuff**
- **Verify the implementation**: scanning finds generic **configuration** and missing **patch** problems

OWASP TOP 10

- A10: Un-validated Redirects and Forwards
- A9: Using Known Vulnerable Components
- A8: Cross-Site Request Forgery (CSRF)
- A7: Missing Function Level Action Control
- A6: Sensitive Data Exposure
- A5: Security Misconfiguration
- **A4: Insecure Direct Object Reference**

A4: Attack Scenarios

- **Checking online how you did in an exam**
  - http://universi.ty/marks?id=i99a**19**

- **Checking how your fellow students did**
  - http://universi.ty/marks?id=i99a**01**
  - http://universi.ty/marks?id=i99a**02**
  - ..
  - http://universi.ty/marks?id=i99a**20**

A4: Risks

- **Common mistakes**
  - Only listing the **“authorized” objects** for the **current user.**
  - **Hiding the object** references in **hidden fields.**
  - **Not enforcing** these restrictions on **server side ~ Presentation Layer Access Control**.
  - Attacker **tampers** those **parameter value**.

A4: Impact

- A**ccess unauthorized files or data** (like registry keys)**.**

A4: Prevent

- **Eliminate the Direct Object References**
  - Eliminate with temporary mapping value (ESAPI **AccessReferenceMap**).
- **Validate the Direct Object Reference**
  - Verify **parameter value format (**Query string constraints)**.**
  - Verify user **authorization** to **access target object (**Data Access Restriction).
  - \ **bject**
  (

OWASP TOP 10

- A10: Un-validated Redirects and Forwards
- A9: Using Known Vulnerable Components
- A8: Cross-Site Request Forgery (CSRF)
- A7: Missing Function Level Action Control
- A6: Sensitive Data Exposure
- A5: Security Misconfiguration
- A4: Insecure Direct Object Reference
- **A3: Cross-Site Scripting (XSS)**

A3: Risks

- **Sends malicious code** to an **innocent user's browser**.
- **Malicious code** might be
  - Reflected from **web input.**
  - Stored in **database**.
  - Sent directly into **rich JavaScript client**.

A3: Attack Scenarios

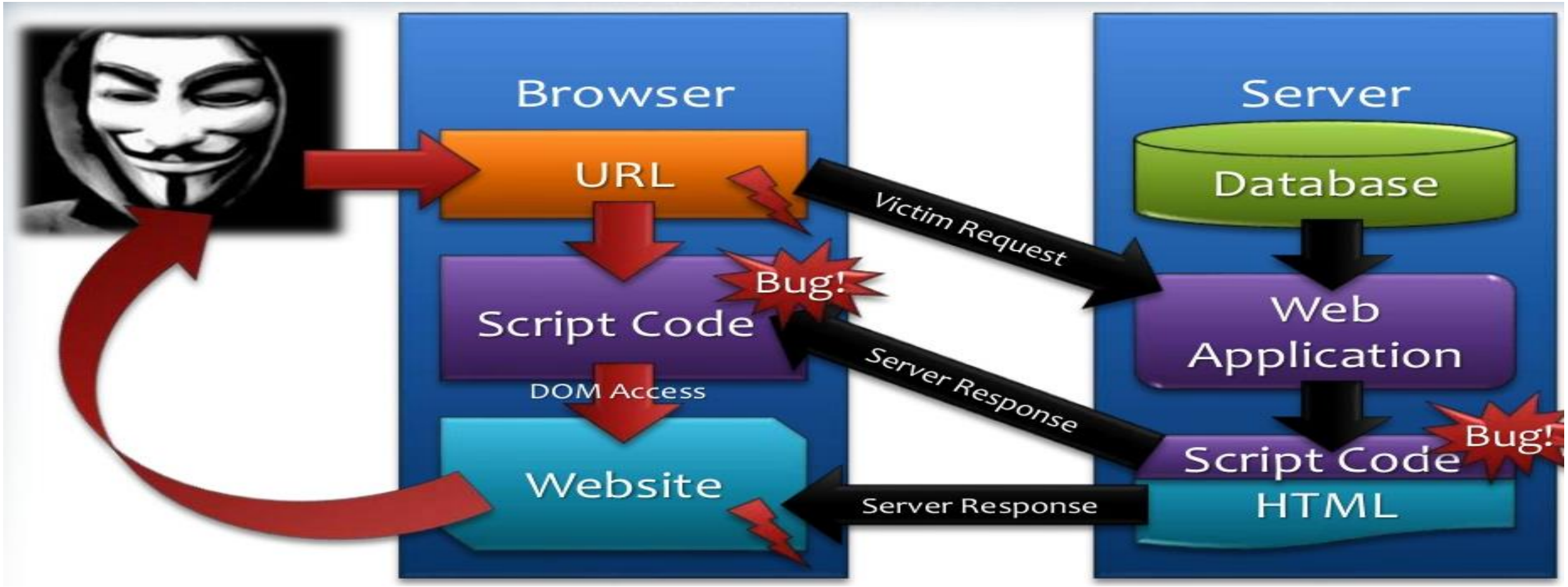
- Reflected XSS

A3: Attack Scenarios (cont.)

- Persistent XSS

A3: Attack Scenarios (cont.)

- Local XSS



A3: Impact

- Steal **sensitive data**
- **Rewrite web page**
- **Redirect** user to **phishing or malware site**

A3: Attack Patterns

- **Simple Patterns**
  - <script>javascript:alert(‘XSS’);</script>
  - <IMG SRC=javascript:alert(‘XSS’)>
  - <IFRAME SRC=“javascript:alert(‘XSS’);”></IFRAME>
- **Masked / Evasion Patterns**
  - <IMG SRC=javascript:alert("XSS")>
  - ";!--"<XSS>=&{()}
  - <IMG """"><SCRIPT>alert("XSS")</SCRIPT>">
  - <IMG SRC="javascript:alert('XSS');">
  - <IMG SRC="jav	ascript:alert('XSS');">

A3: Attack Patterns (cont.)

- **Masked / Evasion Patterns** (cont.)
  - <DIV STYLE="background-image:\0075\0072\006C\0028'\006a\0061\0076\0061\0073\0063\0072\0069\0070\0074\003a\0061\006c\0065\0072\0074\0028.1027\0058.1053\0053\0027\0029'\0029">
  - <b onmouseover=alert('Wufff!')>click me!
  - <img src="http://url.to.file.which/not.exist" onerror=alert(document.cookie);>
  - …

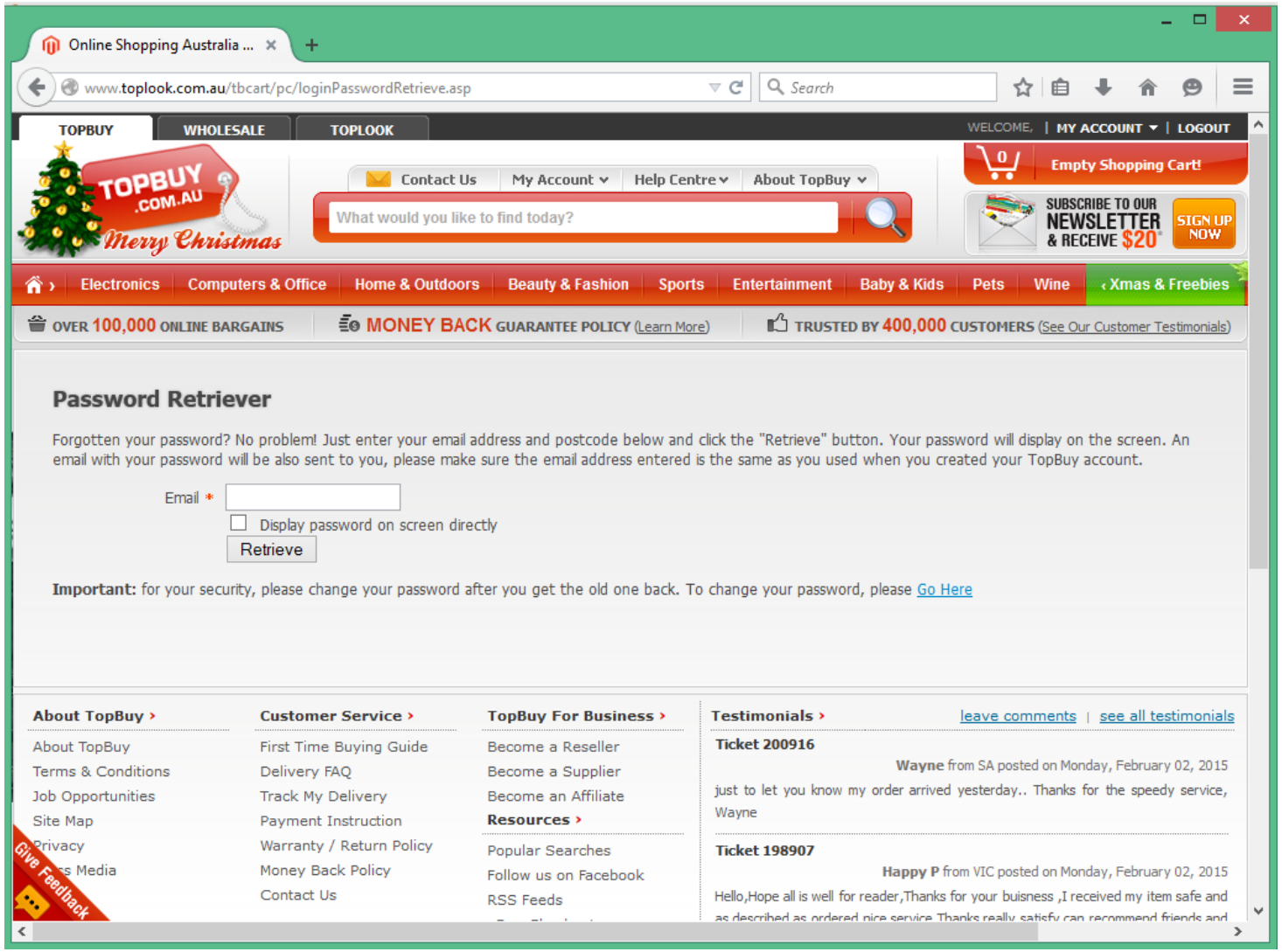
A3: Prevent

- **Eliminate XSS**: Don’t include **user supplied input** in your output.
- **Defend against XSS**: **Output Encode** all user supplied input (**OWASP Enterprise Security API Library** in Java, **Data Annotation** attributes in .NET).
- **White List Input Validation** on user input.
- **HTML Sanitizer** for larger user supplied HTML chunks

OWASP TOP 10

- A10: Un-validated Redirects and Forwards
- A9: Using Known Vulnerable Components
- A8: Cross-Site Request Forgery (CSRF)
- A7: Missing Function Level Action Control
- A6: Sensitive Data Exposure
- A5: Security Misconfiguration
- A4: Insecure Direct Object Reference
- A3: Cross-Site Scripting (XSS)
- **A2: Broken Authentication & Session Management**

A2: What wrong with it?



A2: Risks (SESSIONID generator) *

```
h5kek4z9ha1rtrf
gj7513k7hb15rtr
l8165k45hc1rw7i
p05jrj53hd1i039
5urltda1he1bn46
j51e97h9hf2yq3h
po9531d7hg2awi9
t6zhj2n5hh27bn0
iu345r53hi2aw34
o0z43411hj2njkl
9por42o9hk3dfrz
...
```

Pattern

- 9,7,5,3,1,9,7,5,3,1,9…
- h,h,h,h,h,h,h,h,h,h,h,…
- a,b,c,d,e,f,g,h,i,j,k,…
- 1,1,1,1,1,2,2,2,2,2,3,…

A2: Impact

- HTTP is a “**stateless**” protocol
- Session Management flaws
- Beware the side-doors
- Typical Impact: User accounts compromised or user sessions hijacked.

A2: Prevent (*)

- **Authentication (**simple, centralized and **standardized).**
- **Use standard session ID**.
- **Protect credentials and session ID with SSL/TLS and IPSec protocols.**
- Keep your SSL certificate safe.
- Automatic logout of inactive sessions.
- Never start a login process from an unencrypted page.
- Session IDs and credentials don’t belong into logfiles.

A2: Prevent (cont.)

- Rely on **single authentication mechanism** with **appropriate strength** and number of factors.
- Use strong **supplemental authentication** mechanisms

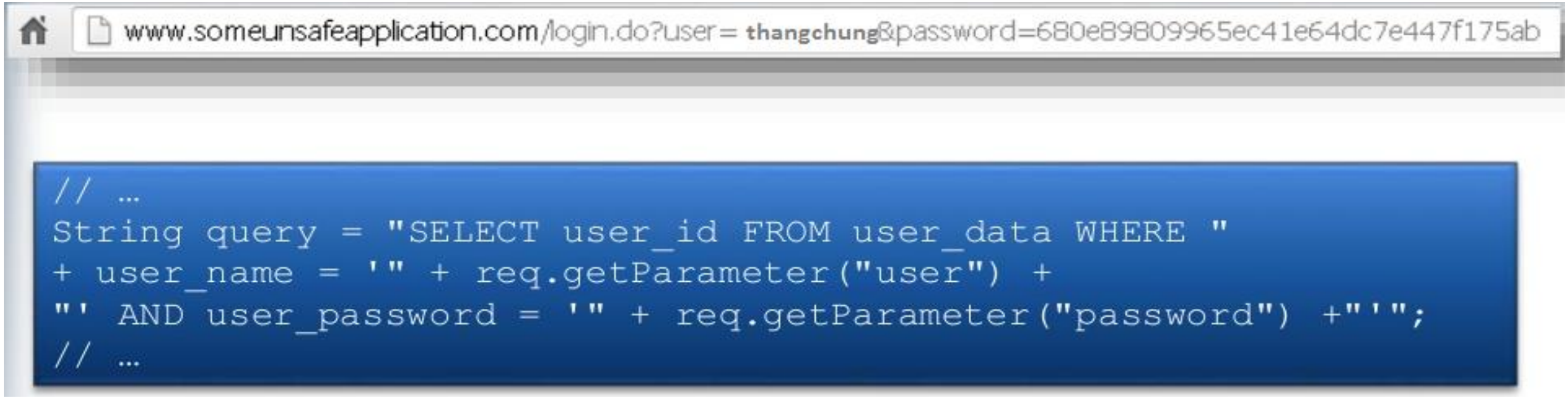
OWASP TOP 10

- A10: Un-validated Redirects and Forwards
- A9: Using Known Vulnerable Components
- A8: Cross-Site Request Forgery (CSRF)
- A7: Missing Function Level Action Control
- A6: Sensitive Data Exposure
- A5: Security Misconfiguration
- A4: Insecure Direct Object Reference
- A3: Cross-Site Scripting (XSS)
- A2: Broken Authentication & Session Management
- **A1: Injection**

A1: Impact

- **Do** something for **tricking** an application into **including unintended commands** in the data sent to an **interpreter**
- Interpreters
  - Take strings and interpret them as commands
  - SQL, OS Shell, LDAP, XPath, ORM, etc…

A1: Attack Scenarios



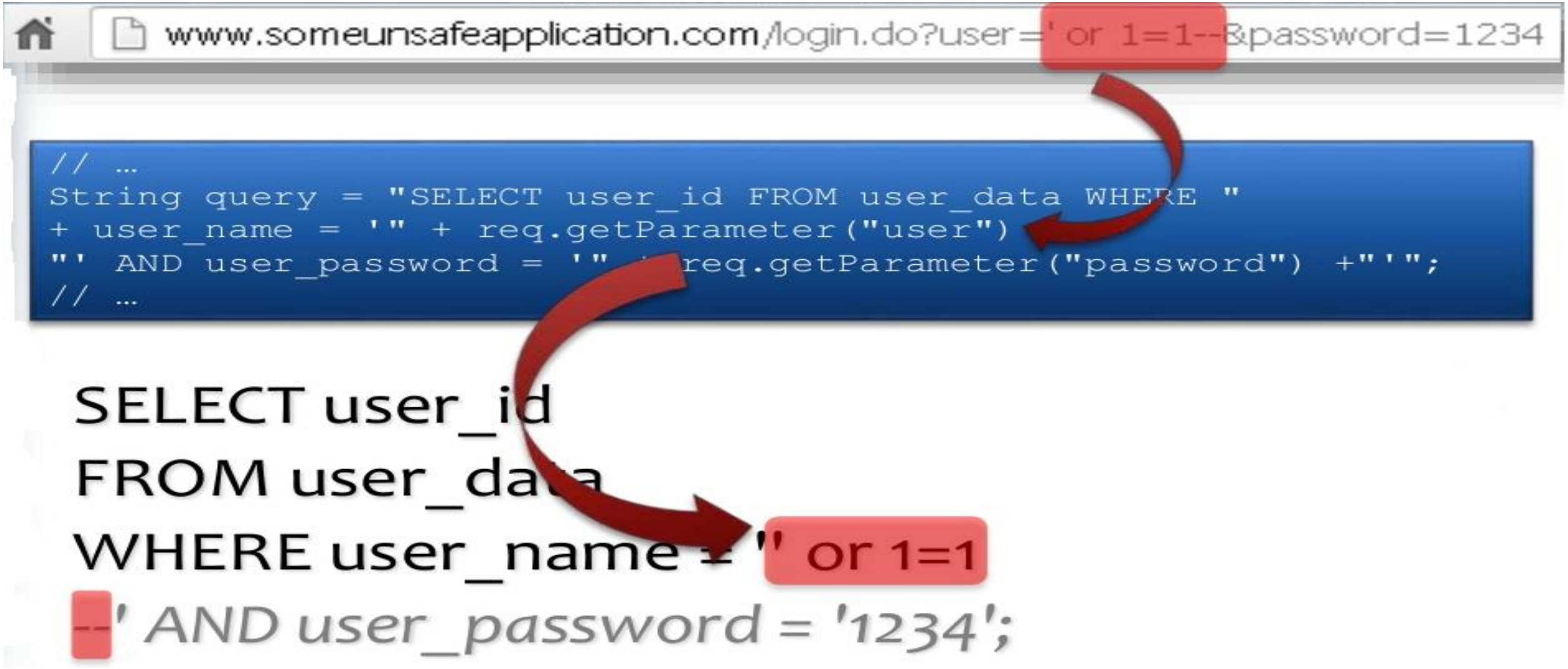
SELECT user_id **FROM** user_data

WHERE user_name = ‘thangchung’ **AND**

user_password=‘680e89809965ec41e64dc7e447f175ab’

-

A1: Attack Scenarios (cont.)



A1: Typical SQL Injection Attack Patterns

- Bypass authentication
  - admin'--
  - admin'#
  - admin'/*
  - ' or 1=1--
  - ' or 1=1#
  - ' or 1=1/*
  - ') or '1'='1
  - ') or ('1'='1

A1: Typical SQL Injection Attack Patterns (cont.)

- Spy out data
  - ‘ UNION SELECT login, password, ‘x’ FROM user—
  - 1 UNION SELECT 1, 1, 1 FROM user --
- Manipulate data
  - ‘;UPDATE user SET type=‘admin’ WHERE id=23;--
- Manipulate the DB server
  - ‘;GO EXEC cmdshell(‘format C’)--

A1: Impact

- Typical Impact
  - **Spy out** or **manipulate** data
  - **Manipulate the DB server** or **access underlying OS**
  - **Bypass authentication** or **gain admin privileges**
- Correlation with Information Leakage
- Blind SQL Injection

A1: Prevent (*)

- **Avoid the Interpreter**
- **Use an interface that supports bind variables**

  (Java.sql.PreparedStatement, Hibernate Parameter Binding, Entity Framework Parameter Binding…)
- **Least Privileges**
- **White List Input Validation** (allow list validation)

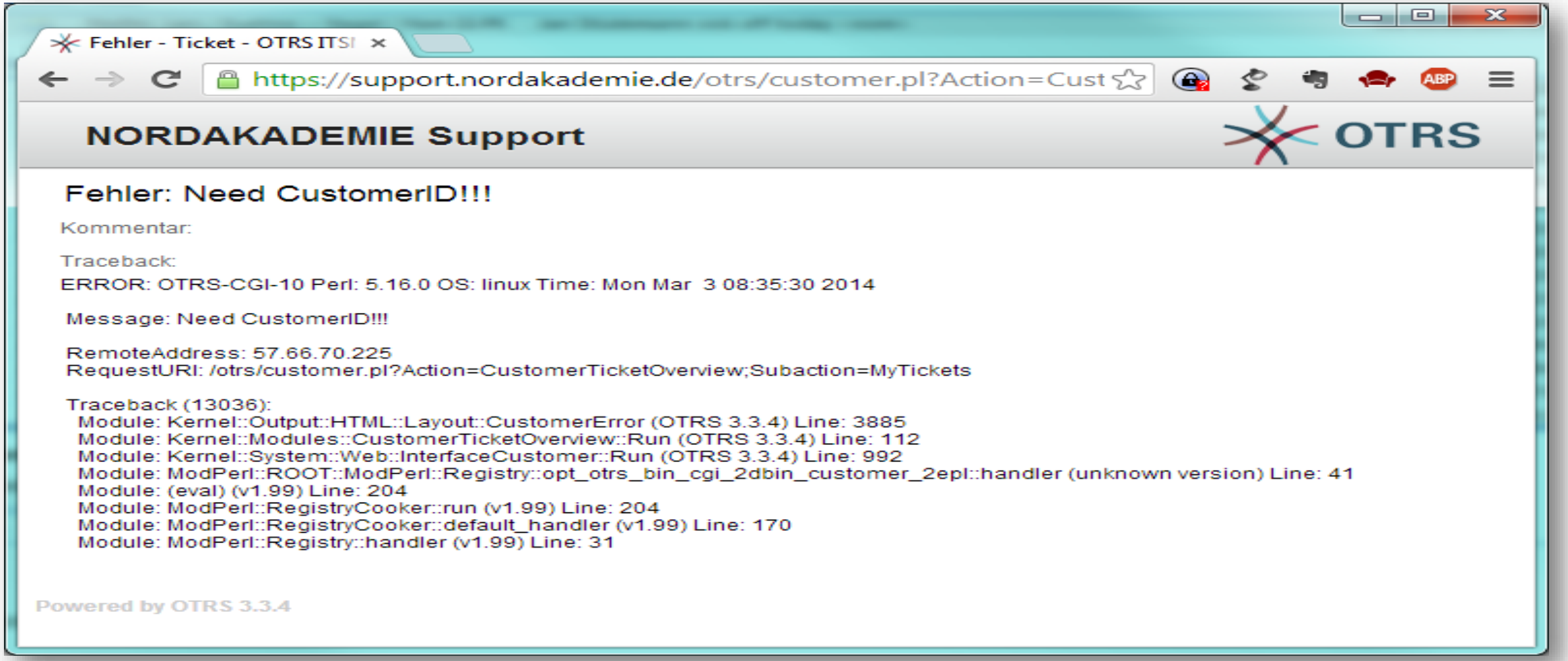
OWASP TOP 10

- A10: Un-validated Redirects and Forwards
- A9: Using Known Vulnerable Components
- A8: Cross-Site Request Forgery (CSRF)
- A7: Missing Function Level Action Control
- A6: Sensitive Data Exposure
- A5: Security Misconfiguration
- A4: Insecure Direct Object Reference
- A3: Cross-Site Scripting (XSS)
- A2: Broken Authentication & Session Management
- A1: Injection
- **Ex-A6/2007: Information Leakage and Improper Error Handling**

Ex-A6/2007: Risk Rating Factors

- **Prevalence**: **Widespread**
- **Impact**: *Minor*

Ex-A6/2007: Information Leakage



Ex-A6/2007: Information Harvest

- **Implementation details**
  - **Server** (OS, Version…).
  - **Programming Language** (Language, Version, VM-Vendor…).
  - **Database** (Oracle, mySQL…) and **details of it** (Version, Schema Names, Table Names, Column Names…).
  - Names and version of used **3rd party libraries.**
- **Other useful information:** Stacktraces, Debugging information, SQL Statements, Password…

Ex-A6/2007: How to prevent it?

- **Common** approach to **exception handling.**
- **Disable** or **limit detailed of error messages.**
- **Secure paths that have multiple outcomes return similar or identical error messages** in roughly the same time.
- **Create a default error handler** (returns an **sanitized error messages**).

Secure Software and Tools Supporting

Secure Software and Tools Supporting

Web Application Risk	Security Utilities (Tools, Services)
A1: Injection	SQL Inject Me and Zed Attack Proxy (ZAP)
A2: Broken Authentication and Session Management	ZAP
A3: Cross-site Scripting (XSS)	ZAP
A4: Insecure Direct Object Reference	HTTP Directory Traversal Scanner, Burp Suite and ZAP
A5: Security Misconfiguration	OpenVAS and WATOBO
A6: Sensitive Data Exposure	Qualys SSL Server Test
A7: Missing Function Level Access Control	OpenVAS

Secure Software and Tools Supporting (cont.)

Web Application Risk	Security Utilities (Tools, Services)
A8: Cross-Site Request Forgery (CSRF)	Tamper Data (Samurai WTF), WebScarab or ZAP
A9: Using Components with Known Vulnerabilities	OpenVAS
A10: Unvalidated Redirects and Forwards	ZAP
Ex-A6/2007: Information Leakage and Improper Error Handling	ZAP

References

- OWASP Top 10 Website
- OWASP TOP 10 -2013 mini book
- OWASP Top 10 for .NET developers mini book
- Bảo mật nhập môn – Phạm Huy Hoàng

NASH TECH
The Power to Innovate

Q&A

THANK YOU

www.nashtechglobal.com